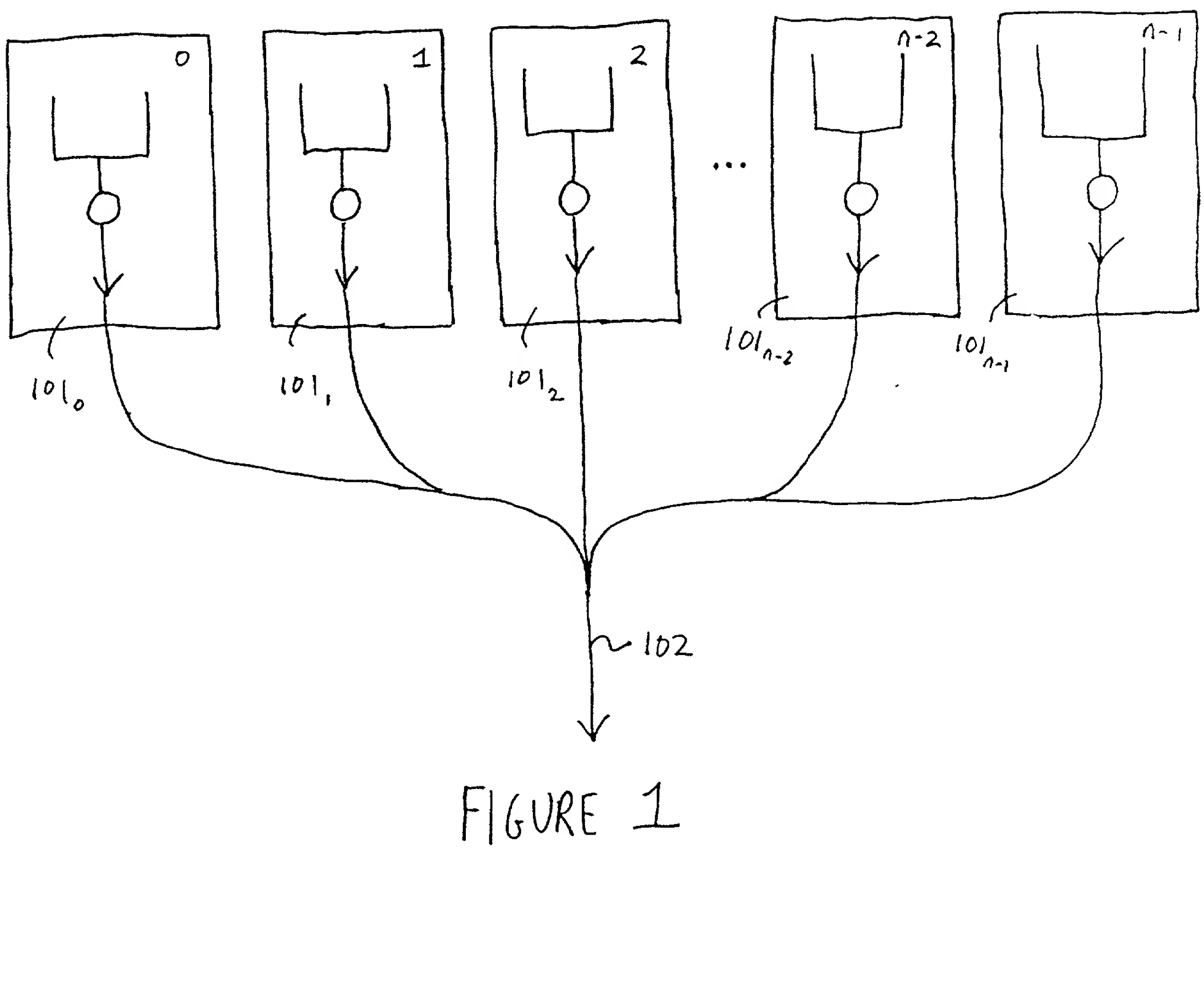


FIGURE 1

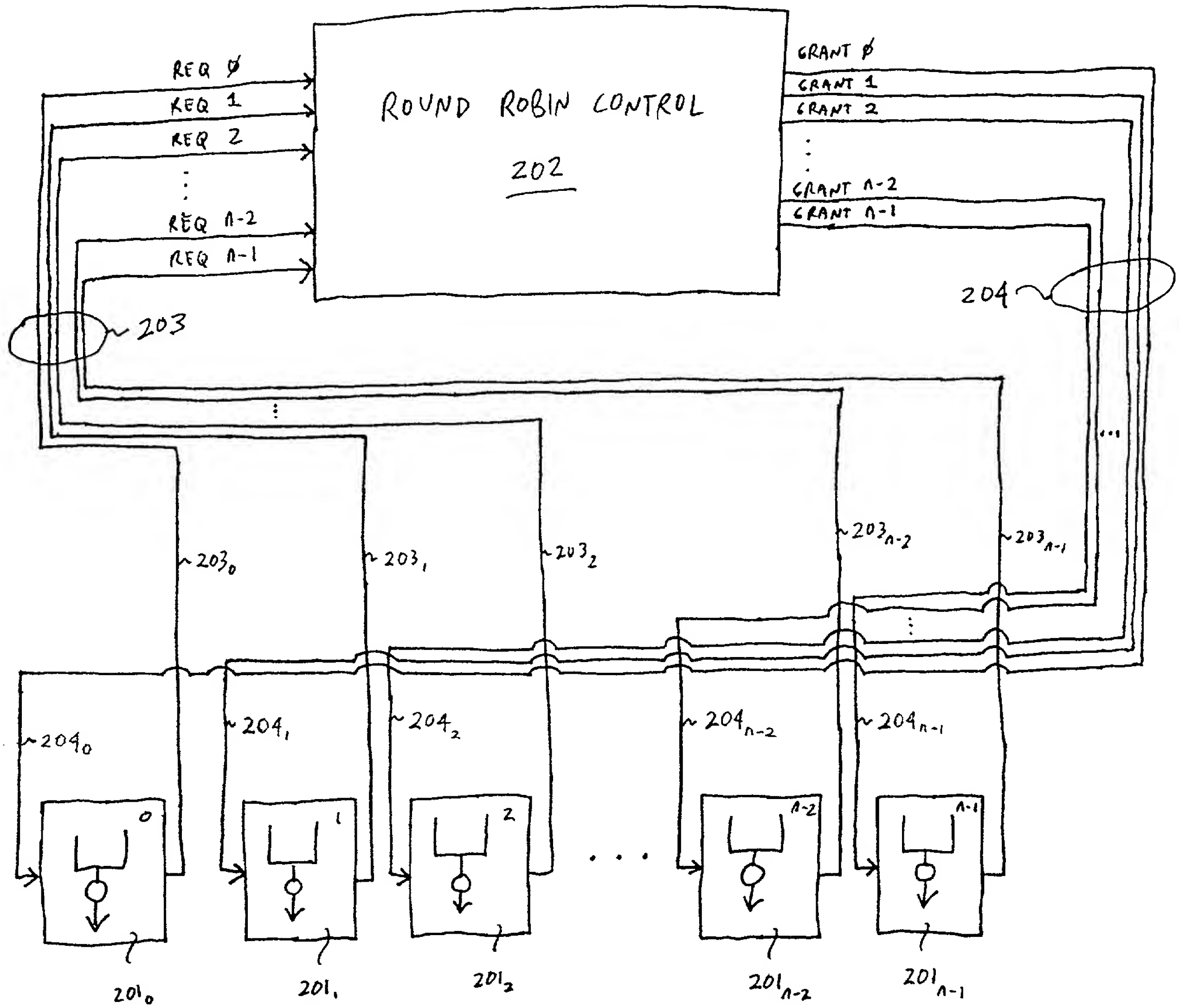

ROUND ROBIN CONTROL
202
REQ Ø
REQ 1
REQ 2
REQ n-2
REQ n-1
GRANT Ø
GRANT 1
GRANT 2
GRANT n-2
GRANT n-1
203
204
203₀
203₁
203₂
203n-2
203n-1
204₀
204₁
204₂
204n-2
204n-1
201₀
201₁
201₂
201n-2
201n-1

FIGURE 2

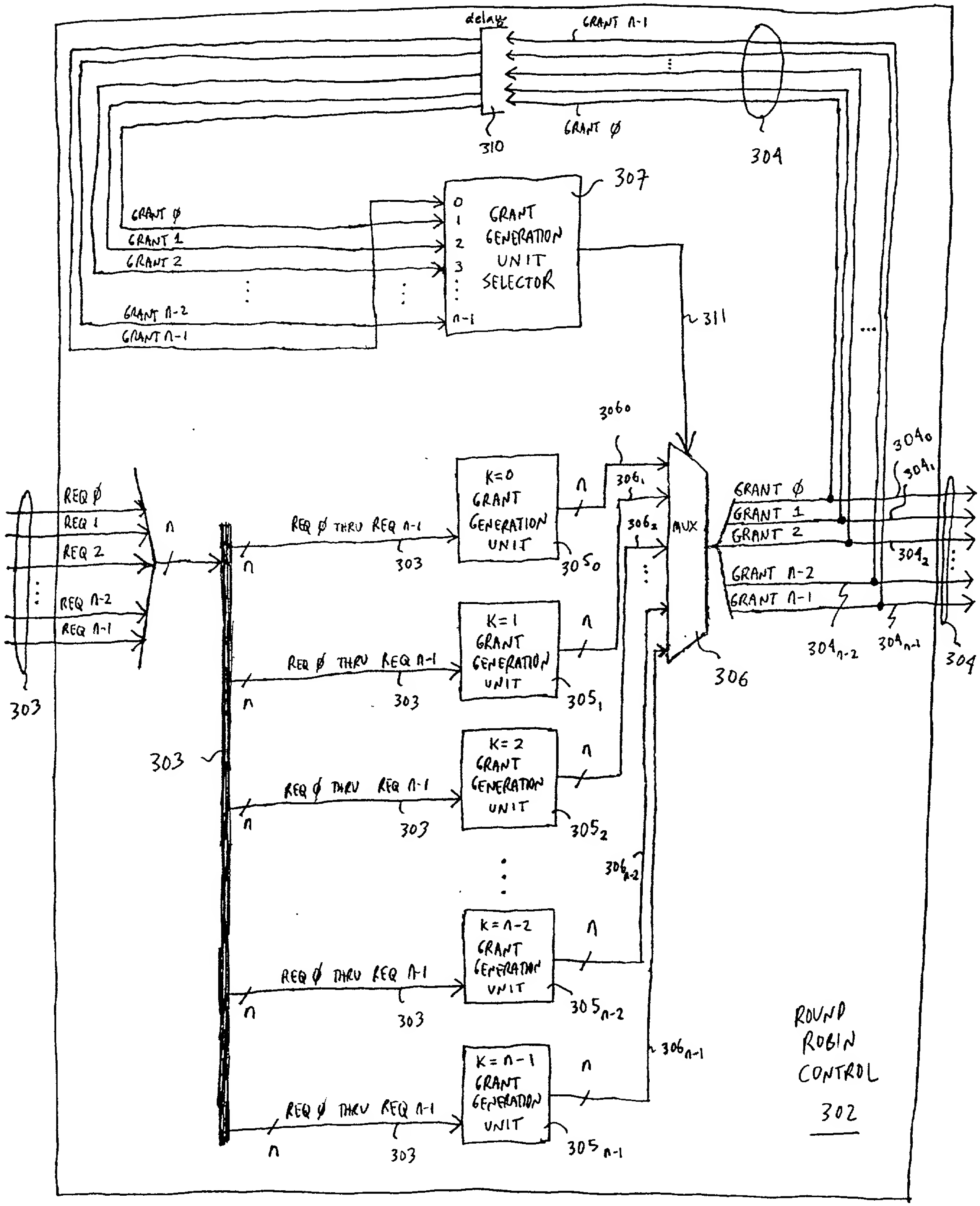

delay
GRANT n-1
GRANT ∅
310
304
307
GRANT GENERATION UNIT SELECTOR
GRANT ∅
GRANT 1
GRANT 2
GRANT n-2
GRANT n-1
311
306₀
306₁
306₂
306ₙ₋₂
306ₙ₋₁
MUX
306
REQ ∅
REQ 1
REQ 2
REQ n-2
REQ n-1
303
REQ ∅ THRU REQ n-1
K=0 GRANT GENERATION UNIT
K=1 GRANT GENERATION UNIT
K=2 GRANT GENERATION UNIT
K=n-2 GRANT GENERATION UNIT
K=n-1 GRANT GENERATION UNIT
305₀
305₁
305₂
305ₙ₋₂
305ₙ₋₁
GRANT ∅
GRANT 1
GRANT 2
GRANT n-2
GRANT n-1
304₀
304₁
304₂
304ₙ₋₂
304ₙ₋₁
ROUND ROBIN CONTROL
302

FIGURE 3a

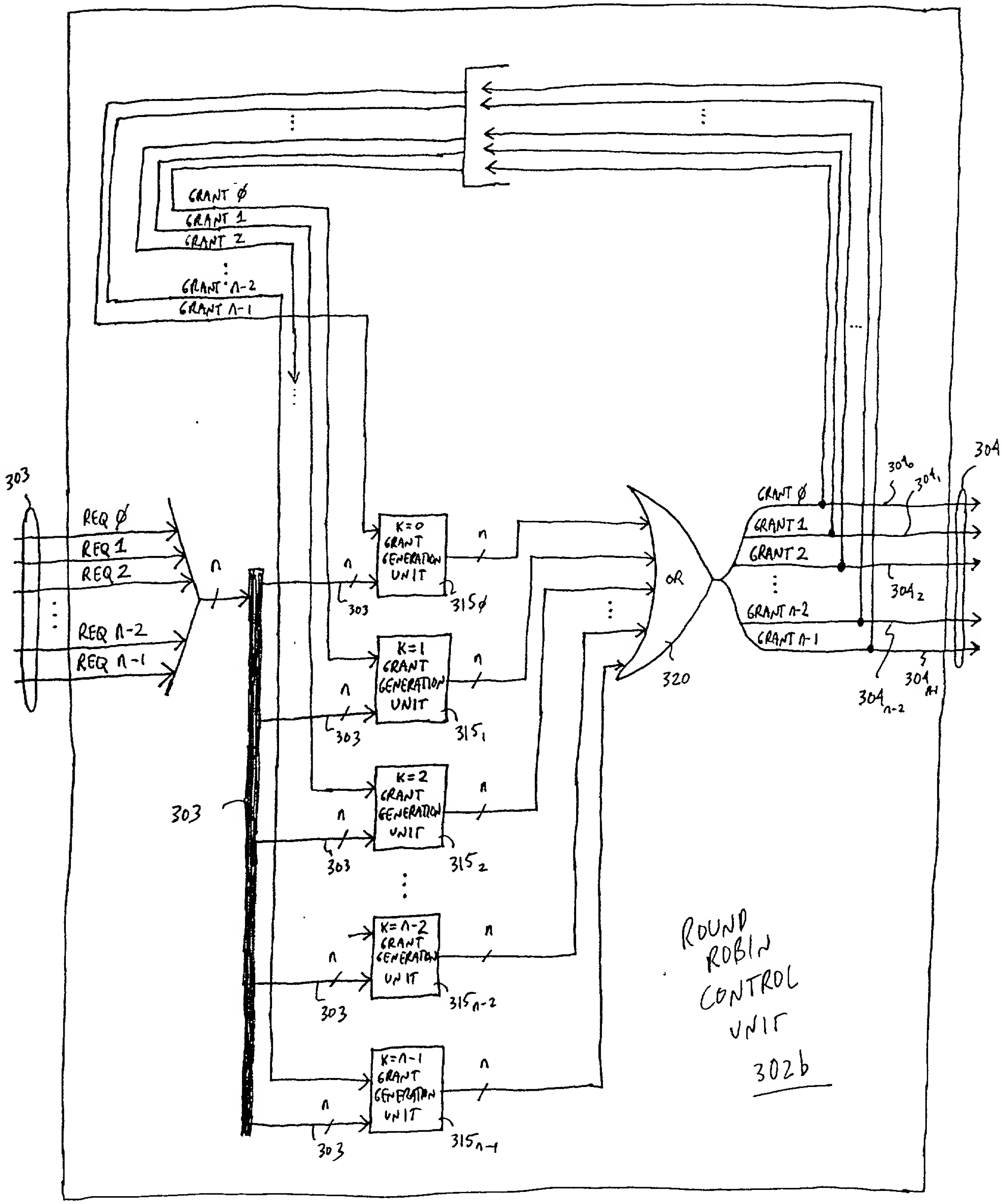

GRANT Ø
GRANT 1
GRANT 2
GRANT n-2
GRANT n-1
REQ Ø
REQ 1
REQ 2
REQ n-2
REQ n-1
303
n
K=0 GRANT GENERATION UNIT
315Ø
K=1 GRANT GENERATION UNIT
315₁
K=2 GRANT GENERATION UNIT
315₂
K=n-2 GRANT GENERATION UNIT
315n-2
K=n-1 GRANT GENERATION UNIT
315n-1
OR
320
304
ROUND ROBIN CONTROL UNIT
302b

FIGURE 3b

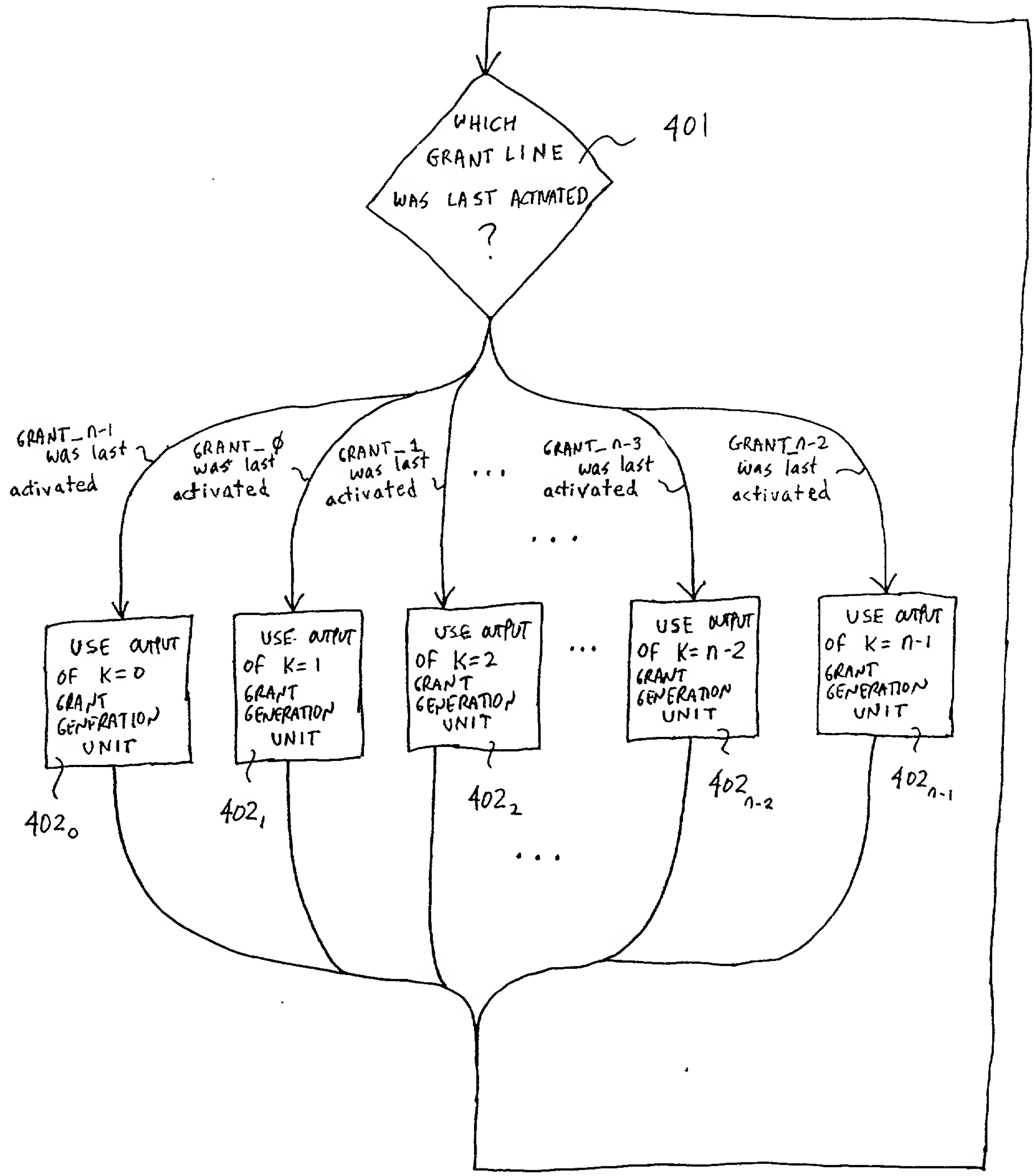

WHICH GRANT LINE WAS LAST ACTIVATED ?
401
GRANT_n-1 was last activated
GRANT_Ø was last activated
GRANT_1 was last activated
GRANT_n-3 was last activated
GRANT_n-2 was last activated
USE OUTPUT OF K=0 GRANT GENERATION UNIT
USE OUTPUT OF K=1 GRANT GENERATION UNIT
USE OUTPUT OF K=2 GRANT GENERATION UNIT
USE OUTPUT OF K=n-2 GRANT GENERATION UNIT
USE OUTPUT OF K=n-1 GRANT GENERATION UNIT
402₀
402₁
402₂
402ₙ₋₂
402ₙ₋₁

FIGURE 4

400

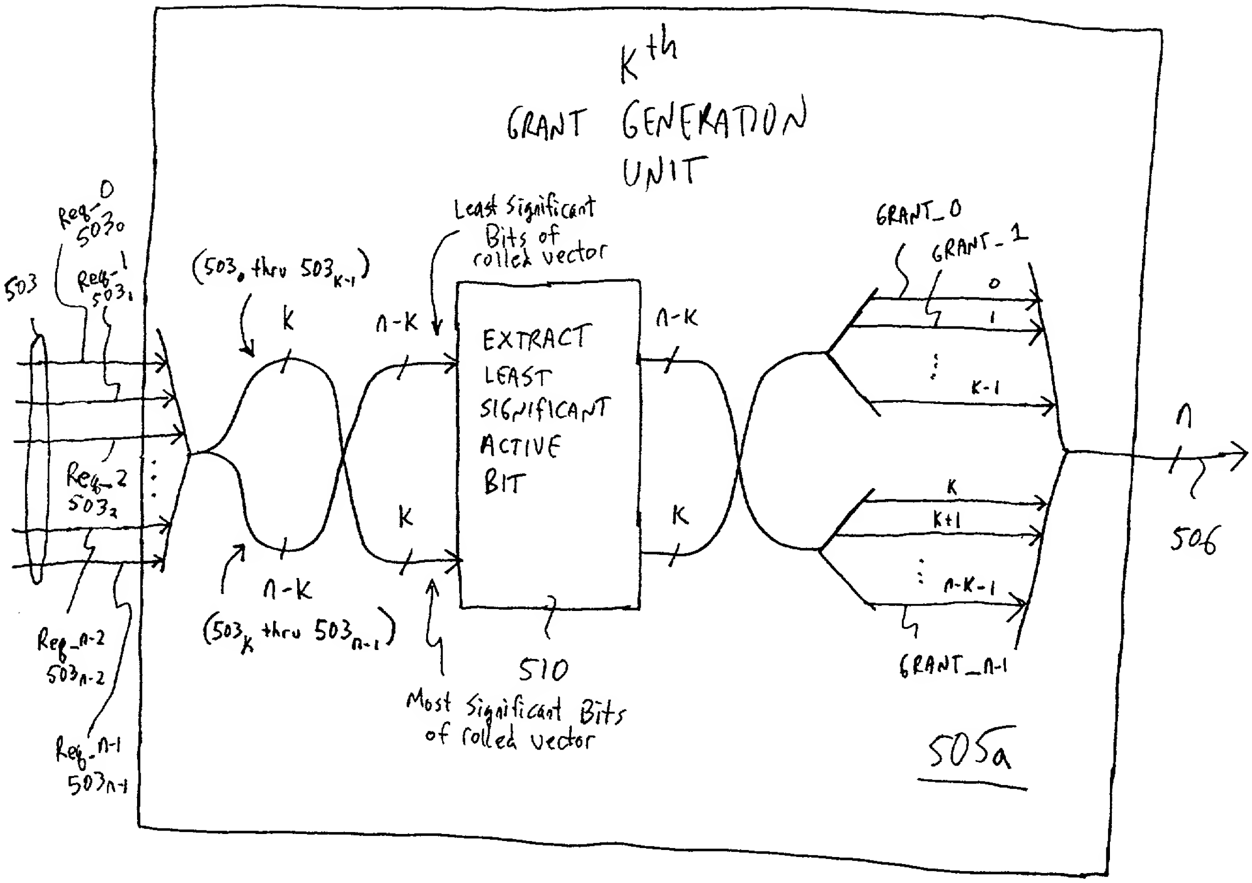
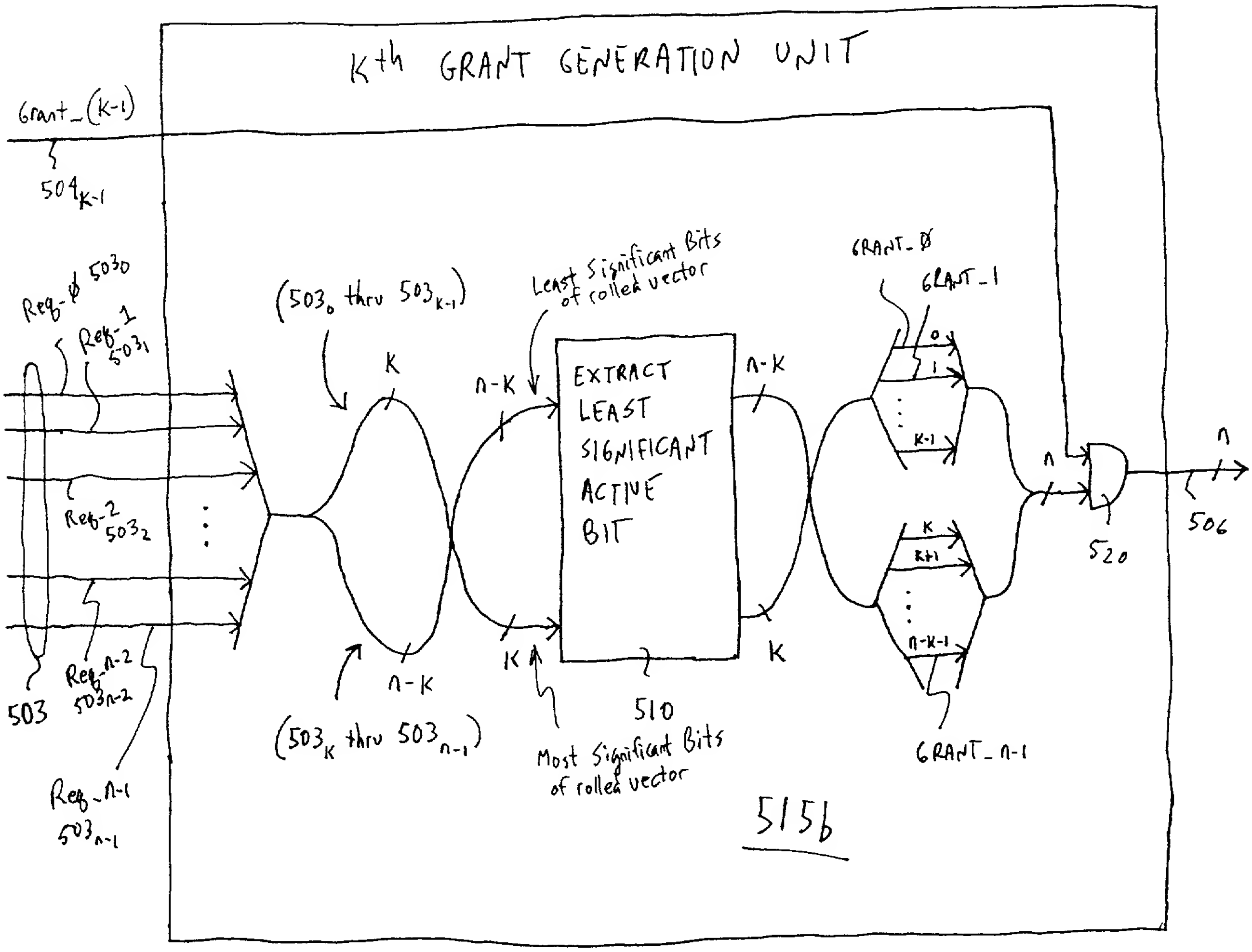

K th
GRANT GENERATION UNIT
Req_0
503₀
Req_1
503₁
503
Req_2
503₂
Req_n-2
503n-2
Req_n-1
503n-1
(503₀ thru 503K-1)
K
n-K
(503K thru 503n-1)
n-K
K
Least Significant Bits of rolled vector
EXTRACT LEAST SIGNIFICANT ACTIVE BIT
Most Significant Bits of rolled vector
510
n-K
K
GRANT_0
GRANT_1
0
1
K-1
K
K+1
n-K-1
GRANT_n-1
n
506
505a

FIGURE 5a



Kth GRANT GENERATION UNIT
Grant_(K-1)
504K-1
Req_Ø 503Ø
Req_1 503₁
Req_2 503₂
Req_n-2 503n-2
Req_n-1 503n-1
503
(503₀ thru 503K-1)
(503K thru 503n-1)
K
n-K
Least Significant Bits of rolled vector
Most Significant Bits of rolled vector
EXTRACT LEAST SIGNIFICANT ACTIVE BIT
510
GRANT_Ø
GRANT_1
GRANT_n-1
K-1
K+1
n-K-1
n
520
506
515b

FIGURE 5b

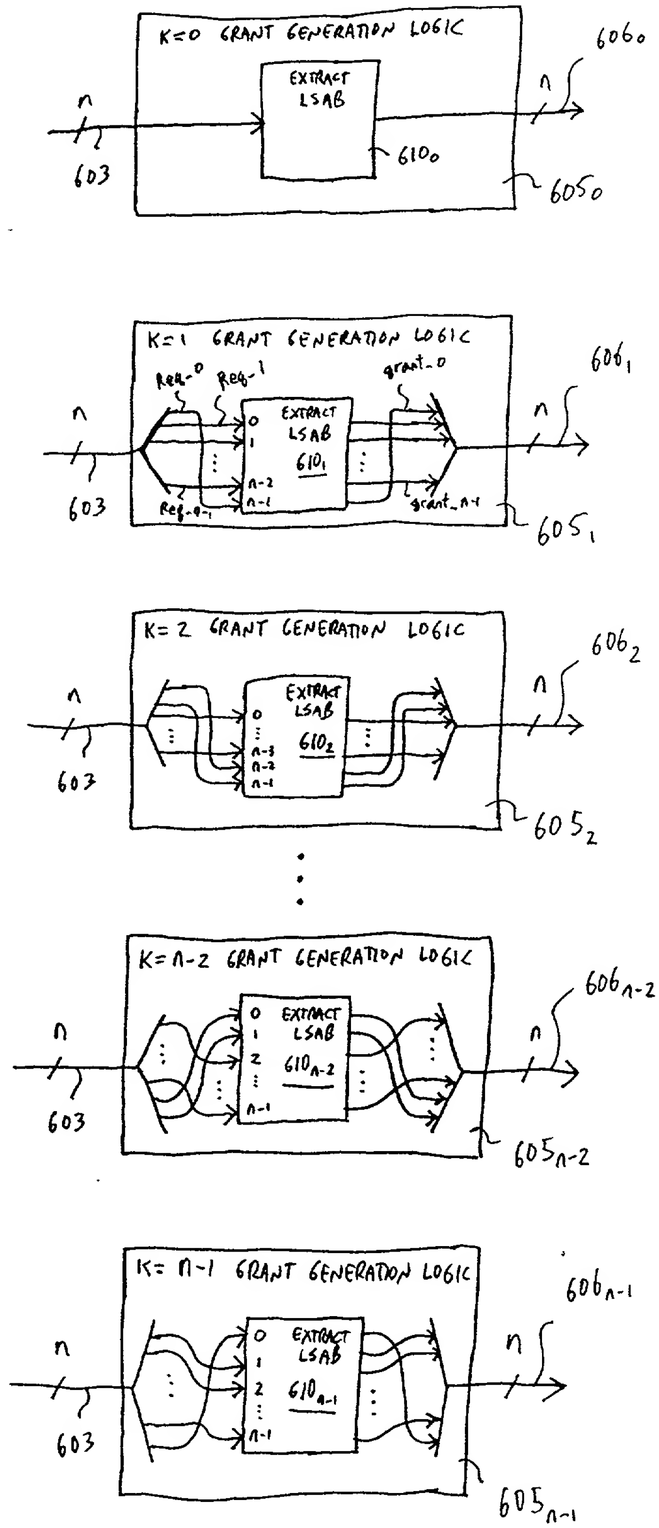

K=0 GRANT GENERATION LOGIC
EXTRACT LSAB
610_0
605_0
606_0
603
n
K=1 GRANT GENERATION LOGIC
Req_0
Req_1
Req_n-1
grant_0
grant_n-1
EXTRACT LSAB
610_1
605_1
606_1
K=2 GRANT GENERATION LOGIC
EXTRACT LSAB
610_2
605_2
606_2
K=n-2 GRANT GENERATION LOGIC
EXTRACT LSAB
610_{n-2}
605_{n-2}
606_{n-2}
K=n-1 GRANT GENERATION LOGIC
EXTRACT LSAB
610_{n-1}
605_{n-1}
606_{n-1}

FIGURE 6

Mats Frannhagen
mfrannhagen@turinnetworks.com

```
module rr (dataIn, state, dataOut) /* synthesis syn_hier = "flatten,remove" */;

input  [19:0]      dataIn;
input  [19:0]      state;
output [19:0]      dataOut;

wire [19:0]            dataOut0, dataOut1, dataOut2, dataOut3, dataOut4,
                       dataOut5, dataOut6, dataOut7, dataOut8, dataOut9,
                       dataOut10, dataOut11, dataOut12, dataOut13, dataOut14,
                       dataOut15, dataOut16, dataOut17, dataOut18, dataOut19;

prio prio0 (.dataIn(dataIn), .en(state[19]),
            .dataOut(dataOut0));

prio prio1 (.dataIn({dataIn[0], dataIn[19:1]}), .en(state[0]),
            .dataOut({dataOut1[0], dataOut1[19:1]}));

prio prio2 (.dataIn({dataIn[1:0], dataIn[19:2]}), .en(state[1]),
            .dataOut({dataOut2[1:0], dataOut2[19:2]}));

prio prio3 (.dataIn({dataIn[2:0], dataIn[19:3]}), .en(state[2]),
            .dataOut({dataOut3[2:0], dataOut3[19:3]}));

prio prio4 (.dataIn({dataIn[3:0], dataIn[19:4]}), .en(state[3]),
            .dataOut({dataOut4[3:0], dataOut4[19:4]}));

prio prio5 (.dataIn({dataIn[4:0], dataIn[19:5]}), .en(state[4]),
            .dataOut({dataOut5[4:0], dataOut5[19:5]}));

prio prio6 (.dataIn({dataIn[5:0], dataIn[19:6]}), .en(state[5]),
            .dataOut({dataOut6[5:0], dataOut6[19:6]}));

prio prio7 (.dataIn({dataIn[6:0], dataIn[19:7]}), .en(state[6]),
            .dataOut({dataOut7[6:0], dataOut7[19:7]}));

prio prio8 (.dataIn({dataIn[7:0], dataIn[19:8]}), .en(state[7]),
            .dataOut({dataOut8[7:0], dataOut8[19:8]}));

prio prio9 (.dataIn({dataIn[8:0], dataIn[19:9]}), .en(state[8]),
            .dataOut({dataOut9[8:0], dataOut9[19:9]}));

prio prio10 (.dataIn({dataIn[9:0], dataIn[19:10]}), .en(state[9]),
             .dataOut({dataOut10[9:0], dataOut10[19:10]}));

prio prio11 (.dataIn({dataIn[10:0], dataIn[19:11]}), .en(state[10]),
             .dataOut({dataOut11[10:0], dataOut11[19:11]}));

prio prio12 (.dataIn({dataIn[11:0], dataIn[19:12]}), .en(state[11]),
             .dataOut({dataOut12[11:0], dataOut12[19:12]}));

prio prio13 (.dataIn({dataIn[12:0], dataIn[19:13]}), .en(state[12]),
             .dataOut({dataOut13[12:0], dataOut13[19:13]}));

prio prio14 (.dataIn({dataIn[13:0], dataIn[19:14]}), .en(state[13]),
             .dataOut({dataOut14[13:0], dataOut14[19:14]}));

prio prio15 (.dataIn({dataIn[14:0], dataIn[19:15]}), .en(state[14]),
             .dataOut({dataOut15[14:0], dataOut15[19:15]}));

prio prio16 (.dataIn({dataIn[15:0], dataIn[19:16]}), .en(state[15]),
             .dataOut({dataOut16[15:0], dataOut16[19:16]}));

prio prio17 (.dataIn({dataIn[16:0], dataIn[19:17]}), .en(state[16]),
             .dataOut({dataOut17[16:0], dataOut17[19:17]}));

prio prio18 (.dataIn({dataIn[17:0], dataIn[19:18]}), .en(state[17]),
             .dataOut({dataOut18[17:0], dataOut18[19:18]}));

prio prio19 (.dataIn({dataIn[18:0], dataIn[19]}), .en(state[18]),
             .dataOut({dataOut19[18:0], dataOut19[19]}));

assign dataOut = ((dataOut0 | dataOut1 | dataOut2 | dataOut3) |
                  (dataOut4 | dataOut5 | dataOut6 | dataOut7) |
                  (dataOut8 | dataOut9 | dataOut10 | dataOut11) |
                  (dataOut12 | dataOut13 | dataOut14 | dataOut15) |
                  (dataOut16 | dataOut17 | dataOut18 | dataOut19));
endmodule // rr
```

702

703

FIGURE 7

700

Figure 4. The "round robin" top level Verilog module for N=20

Mats Frannhagen
mfrannhagen@turinnetworks.com

FIGURE 8

```
module prio (dataIn, en, dataOut);
input [19:0]     dataIn;
input            en;
output [19:0]    dataOut;

reg [19:0]       i_dataOut0;


always @(/*AUTOSENSE*/dataIn) begin
  i_dataOut0 = 20'd0;
  if (dataIn[0])          i_dataOut0 = 20'h00001;
  else if (dataIn[1])     i_dataOut0 = 20'h00002;
  else if (dataIn[2])     i_dataOut0 = 20'h00004;
  else if (dataIn[3])     i_dataOut0 = 20'h00008;
  else if (dataIn[4])     i_dataOut0 = 20'h00010;
  else if (dataIn[5])     i_dataOut0 = 20'h00020;
  else if (dataIn[6])     i_dataOut0 = 20'h00040;
  else if (dataIn[7])     i_dataOut0 = 20'h00080;
  else if (dataIn[8])     i_dataOut0 = 20'h00100;
  else if (dataIn[9])     i_dataOut0 = 20'h00200;    801
  else if (dataIn[10])    i_dataOut0 = 20'h00400;
  else if (dataIn[11])    i_dataOut0 = 20'h00800;
  else if (dataIn[12])    i_dataOut0 = 20'h01000;
  else if (dataIn[13])    i_dataOut0 = 20'h02000;
  else if (dataIn[14])    i_dataOut0 = 20'h04000;
  else if (dataIn[15])    i_dataOut0 = 20'h08000;
  else if (dataIn[16])    i_dataOut0 = 20'h10000;
  else if (dataIn[17])    i_dataOut0 = 20'h20000;
  else if (dataIn[18])    i_dataOut0 = 20'h40000;
  else if (dataIn[19])    i_dataOut0 = 20'h80000;
end

assign dataOut = {20{en}} & i_dataOut0;

endmodule // prio
```

Figure 5. The basic "brute force" verilog implementation of the "prio" module for N=20. This generates the smallest area but is slower than the alternative implementation shown next.

Mats Frannhagen
mfrannhagen@turinnetworks.com

FIGURE 9

```
module prio (dataIn, en, dataOut) /* synthesis syn_hier = "flatten,remove" */;
input [19:0]     dataIn;
input            en;
output [19:0]    dataOut;

reg [4:0]        i_dataOut0;
reg [4:0]        i_dataOut1;
reg [4:0]        i_dataOut2;
reg [4:0]        i_dataOut3;
wire [9:0]       i_dataOut4;
wire [9:0]       i_dataOut5;

wire             muxCtl1;
wire             muxCtl2;
wire             muxCtl3;

// Calc in parallel
assign muxCtl1 = |dataIn[4:0];
assign muxCtl2 = |dataIn[14:10];
assign muxCtl3 = |dataIn[9:5] | muxCtl1;

always @(/*AUTOSENSE*/dataIn) begin
  i_dataOut0 = 5'd0;
  if (dataIn[0])          i_dataOut0 = 5'h01;
  else if (dataIn[1])     i_dataOut0 = 5'h02;
  else if (dataIn[2])     i_dataOut0 = 5'h04;
  else if (dataIn[3])     i_dataOut0 = 5'h08;
  else if (dataIn[4])     i_dataOut0 = 5'h10;
end // always @ (...
```

901

```
always @(/*AUTOSENSE*/dataIn) begin
  i_dataOut1 = 5'd0;
  if (dataIn[5])          i_dataOut1 = 5'h01;
  else if (dataIn[6])     i_dataOut1 = 5'h02;
  else if (dataIn[7])     i_dataOut1 = 5'h04;
  else if (dataIn[8])     i_dataOut1 = 5'h08;
  else if (dataIn[9])     i_dataOut1 = 5'h10;
end // always @ (...
```

902

```
always @(/*AUTOSENSE*/dataIn) begin
  i_dataOut2 = 5'd0;
  if (dataIn[10])         i_dataOut2 = 5'h01;
  else if (dataIn[11])    i_dataOut2 = 5'h02;
  else if (dataIn[12])    i_dataOut2 = 5'h04;
  else if (dataIn[13])    i_dataOut2 = 5'h08;
  else if (dataIn[14])    i_dataOut2 = 5'h10;
end // always @ (...
```

903

```
always @(/*AUTOSENSE*/dataIn) begin
  i_dataOut3 = 5'd0;
  if (dataIn[15])         i_dataOut3 = 5'h01;
  else if (dataIn[16])    i_dataOut3 = 5'h02;
  else if (dataIn[17])    i_dataOut3 = 5'h04;
  else if (dataIn[18])    i_dataOut3 = 5'h08;
  else if (dataIn[19])    i_dataOut3 = 5'h10;
end // always @ (...
```

904

905

```
// "Mux" data out
assign i_dataOut4 = {i_dataOut1 & {5{~muxCtl1}}, i_dataOut0 & {5{muxCtl1}}};
assign i_dataOut5 = {i_dataOut3 & {5{~muxCtl2}}, i_dataOut2 & {5{muxCtl2}}};
assign dataOut = {i_dataOut5 & {10{en & ~muxCtl3}}, i_dataOut4 & {10{en & muxCtl3}}};

endmodule // prio
```

Figure 6. Alternative "prio" module Verilog implementation.